CrowdForge: Crowdsourcing Complex Work

Aniket Kittur *Boris Smus* *Susheel Khamkar* *Robert E. Kraut*
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213
nkittur@cs.cmu.edu, boris.smus@gmail.com, susheelkhamkar2@gmail.com, robert.kraut@cs.cmu.edu

ABSTRACT
Micro-task markets such as Amazon's Mechanical Turk represent a new paradigm for accomplishing work, in which employers can tap into a large population of workers around the globe to accomplish tasks in a fraction of the time and money of more traditional methods. However, such markets have been primarily used for simple, independent tasks, such as labeling an image or judging the relevance of a search result. Here we present a general purpose framework for accomplishing complex and interdependent tasks using micro-task markets. We describe our framework, a web-based prototype, and case studies on article writing, decision making, and science journalism that demonstrate the benefits and limitations of the approach.

ACM Classification: H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General terms: Algorithms, Design, Economics, Human Factors.

Keywords: crowdsourcing, Mechanical Turk, human computation, distributed processing, MapReduce, coordination.

INTRODUCTION
Crowdsourcing has become a powerful mechanism for accomplishing work online. Hundreds of thousands of volunteers have completed tasks including classifying craters on planetary surfaces (clickworkers.arc.nasa.gov), deciphering scanned text (recaptcha.net), and discovering new galaxies (galaxyzoo.org). Crowdsourcing has succeeded as a commercial strategy for accomplishing work as well, with companies accomplishing work ranging from crowdsourcing t-shirt designs (Threadless) to research and development (Innocentive).

One of the most interesting developments is the creation of general-purpose markets for crowdsourcing diverse tasks. For example, in Amazon's Mechanical Turk (MTurk), tasks range from labeling images with keywords to judging the relevance of search results to transcribing podcasts. Such "micro-task" markets typically involve short tasks (ranging from a few seconds to a few minutes) which users self-select and complete for monetary gain (typically from 1-10 cents per task). These markets represent the potential for accomplishing work in a fraction of the time and money required by more traditional methods [5][14][22][25].

UIST'11, October 16–19, 2011, Santa Barbara, CA, USA.

However, the types of tasks accomplished through MTurk have typically been limited to those that are low in complexity, independent, and require little time and cognitive effort to complete. The typical task on MTurk is a self-contained, simple, repetitive, and short one, requiring little specialized skill and often paying less than minimum wage. Amazon calls its tasks HITs, for human intelligence tasks. During February 2010, we scraped descriptions of 13,449 HIT groups posted to Mechanical Turk. The modal HIT paid $0.03 US. Examples of typical tasks include identifying objects in a photo or video, de-duplicating data, transcribing audio recordings, or researching data details, with many tasks taking only a fraction of a minute to complete.

In contrast to the typical tasks posted on Mechanical Turk, much of the work required in many real-world work organizations and even many temporary employment assignments is often much more complex, interdependent, and requires significant time and cognitive effort [18]. They require substantially more coordination among co-workers than do the simple, independent tasks seen on micro-task markets. The impact of micro-task markets would be substantially greater if they could also be applied to more complex and interdependent tasks.

Consider for example the task of writing a short article about a locale, a newspaper, a travel guide, or a corporate retreat or an encyclopedia. This is a complex and highly interrelated task that involves many subtasks, such as deciding on the scope and structure of the article, finding and collecting relevant information, writing the narrative, taking pictures, laying out the document and editing the final copy. Furthermore, if more than one person is involved, they need to coordinate in order to avoid redundant work and to make the final product coherent. Many kinds of tasks ranging from researching where to go on vacation to planning a new consumer product to writing software share the properties of being complex and highly interdependent, requiring substantial effort from individual contributors. These challenges are exacerbated in crowdsourcing markets such as MTurk, in which tasks must be simple enough for workers to easily learn and complete and workers have low commitment and unknown expertise and skills.

Here we present the CrowdForge framework and toolkit for crowdsourcing complex work. Our first contribution is conceptualizing a framework for accomplishing complex

and interdependent tasks from many small contributions in crowdsourcing markets. The framework provides a small set of task primitives (partition, map, and reduce) that can be combined and nested to address many crowdsourcing problems. Our second contribution is implementing the framework in a software toolkit, and using this toolkit to investigate the applicability, strengths, and weaknesses of the framework across a diverse range of tasks.

MECHANICAL TURK

Amazon's Mechanical Turk is a "marketplace for work that requires human intelligence" in which employers post generally small and self-contained tasks that workers across the globe can complete for monetary reward. MTurk encompasses a large and heterogeneous pool of tasks and workers; Amazon claims over 100,000 workers in 100 different countries, and as of the time of writing there were approximately 80,000 tasks available. There has been an increasing amount of research on Mechanical Turk recently. A large number of studies have examined the usefulness of MTurk as a platform for collecting and evaluating data for applications including machine translation [4] image databases [7], and natural language processing [22]. There have also been studies examining the use of MTurk as a platform for experiments in perception [5][11], judgment and decision making [13], and user interface evaluation [14]. Sorokin & Forsyth used Mechanical Turk to tag images in computer vision research and suggested voting mechanisms to improve quality [25]. Little et al. **Error! Reference source not found.** introduced TurKit, discussed later in this paper, a toolkit for iterative tasks in MTurk.

APPROACH

Our goal is to support the coordination dependencies involved in complex work done using micro-task markets. Most tasks on these markets are simple, self-contained, and independent. The audio transcription tasks posted by Castingwords.com are a rare exception. Castingwords breaks up an audio stream into overlapping segments, and workers

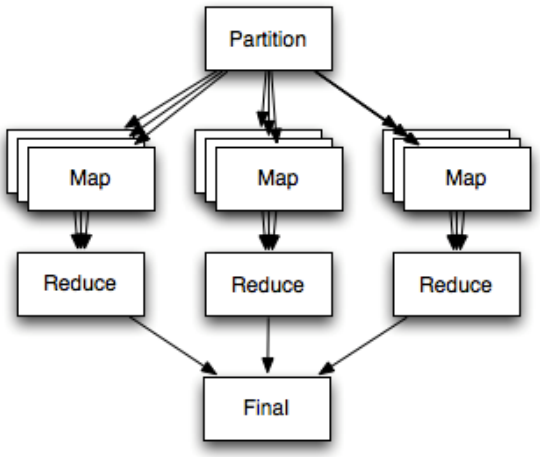

Figure 1: Overview of framework for splitting up and recombining complex human computation tasks.

are employed to generate transcriptions from each audio segment. These transcriptions are then verified by other workers, whose work is later automatically put together into a complete transcription. Unlike the standard micro-market task, the disaggregation of an audio file into smaller transcription tasks and the use of a second wave of workers to verify the work done by the transcriptions involves the producer/consumer dependency identified in [18]. It also provides a simple model for many of the elements of our approach. For example, the transcription task can be broken up into the following elements:

- A pre-specified partition that breaks up the audio into smaller subtasks
- A flow that controls the sequencing of the tasks and transfer of information between them
- A quality control phase that involves verification of one task by another worker
- Automatic aggregation of the results

The TurKit toolkit for MTurk **Error! Reference source not found.** extends some of these elements by enabling task designers to specify iterative flows. Little and colleagues use as an example a text identification in which the results of multiple workers' outputs are voted on and the best sent to new workers, whose work is then voted on, and so forth. This can also be characterized as a producer/consumer dependency [18], with a flow between a production task (text identification) and a quality control task (voting) that serves to aggregate the results.

Our goal is to generalize these elements into a framework that supports the crowdsourcing of highly complex work. Specifically, our framework aims to support:

- Dynamic partitioning so that workers themselves can decide a task partition, with their results in turn generating new subtasks (rather than the task designer needing to fully specify partitions beforehand)
- Multi-level partitions in which a task can be broken up by more than one partition
- Complex flows involving many tasks and workers
- A variety of quality control methods including voting, verification, or merging items
- Intelligent aggregation of results both automatically and by workers.
- A simple method for specifying and managing tasks and flows between tasks

To accomplish these goals our approach draws on concepts from both organizational behavior [18] and distributed computing [1][24]. Malone and Crowston [18] note the general parallels between coordination in human systems and computer systems. Key challenges in both organizations and distributed computing include partitioning work into tasks that can be done in parallel, mapping tasks to workers/processors, managing the dependencies between them, and maintaining quality controls [1][2][18][24]. Crowdsourced labor markets can be viewed as large distri-

buted systems in which each person, such as a worker on Mechanical Turk, is analogous to a computer processor that can solve a task requiring human intelligence. In this way a crowdsourcing market could be seen as a loosely coupled distributed computing system [1]. This suggests that some solutions to managing distributed computing may be profitably applied to crowdsourcing as well.

Our approach builds on the general approach to simplified distributed computing exemplified by systems such as MapReduce [6] of breaking down a complex problem into a sequence of simpler subtasks using a small set of subtask types (e.g., Maps and Reduces) and managing the dependencies between them. MapReduce was inspired by functional programming languages in which a large array of data is processed in parallel through a two step process: first, key/value pairs are each processed to generate a set of intermediate key/value pairs (the Map phase). Next, values with identical intermediate keys are merged (the Reduce phase). Although we use MapReduce terminology as a convenient analog here, our work builds on a larger tradition of simplified distributed computing (e.g., [11][21]). We define three types of subtasks as:

- Partition tasks, in which a larger task is broken down into discrete subtasks
- Map tasks, in which a specified task is processed by one or more workers
- Reduce tasks, in which the results of multiple workers' tasks are merged into a single output

CrowdForge, our prototype system, abstracts away many of the programming details of creating and managing subtasks by treating partition/map/reduce steps as the basic building blocks for distributed process flows, enabling complex tasks to be broken up systematically and dynamically into sequential and parallelizable subtasks.

In partition tasks, workers are asked to create a high level partitioning of the problem, such as creating an outline of an article with section headings or a list of criteria for buying a new car. In CrowdForge the partitioning is made an explicit part of the task itself, with subtasks dynamically created based on the results of the partition step. Importantly, this means that the task designer does not have to know beforehand all of the subtasks that will be generated. Defining the division of labor and subtask design are shifted to the market itself, a key advantage that is novel and unique to human computation.

In map tasks, a specified processing step is applied to each item in the partition. In micro-task markets, these tasks should be simple enough to be completed by a single worker in a short amount of time. For example, a map task for article writing could ask a worker to collect one fact on a given topic in the article's outline. Multiple instances of a map tasks could be instantiated for each partition; e.g., multiple workers could be asked to collect one fact each on the topic in parallel.

Finally, reduce tasks take all the results from a given map task and consolidate them, typically into a single result. In the article writing example, a reduce step might take facts collected for a given topic by many workers and have a worker turn them into a paragraph.

Any of these steps can be iterative. For example, the topic for an article section defined in a first partition can itself be partitioned into subsections. Similarly, the paragraphs returned from one reduction step can in turn be reordered through a second reduction step.

CASE STUDIES

Before describing implementation details of the system we first provide examples and evidence of how the system works in practice through two case studies: article writing and researching a purchase decision.

Article writing

The first complex task we explored was writing an encyclopedia article. Writing an article is a challenging and interdependent task that involves many different subtasks: planning the scope of the article, how it should be structured, finding and filtering information to include, writing up that information, finding and fixing grammar and spelling, and making the article coherent. While there are examples of collaborative writing on the Internet, notably Wikipedia, previous work has shown that the success of harnessing a large group of contributors is often dependent on a small core of leaders that do a large proportion of the work and organize the contributions of others [15][16]. This poses a challenge in micro-task markets where individuals may not be willing to spend the large amount of effort needed to be a leader and may not be able to com-

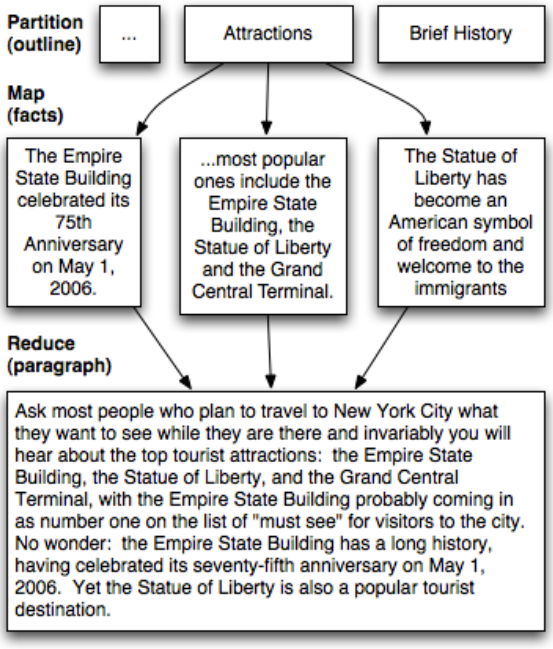

Figure 2: Partial results of a collaborative writing task.

municate with others in order to coordinate or influence their behavior. Furthermore, many of the subtasks involved, such as assembling the relevant information or doing the actual writing, can be time consuming and complex. These characteristics make article writing a challenging but representative test case for our approach.

To solve this problem we created a simple flow consisting of a partition, map, and reduce step (see Figure 2). The partition step asked workers to create an article outline, represented as an array of section headings such as "History" and "Geography". In an environment where workers would complete high effort tasks, the next step might be to have someone write a paragraph for each section. However, the difficulty and time involved in researching and writing a complete paragraph for a heading is a mismatch to the low work capacity of micro-task markets. Thus we broke the task up further, separating the information collection and writing subtasks. Specifically, each section heading from the partition was used to generate map tasks in which multiple workers were asked to submit a single fact about the section (workers were also asked to submit a URL reference to the source of the fact to encourage high quality fact collection).

Next, the reduction step asked other workers to create a paragraph for each section based on the facts collected in the map step. By separating the collection of information and writing into two stages we could significantly decrease the cost of each stage, making the task more suitable for micro-task workers. In addition, we benefit from other effects such being able to collect more diverse information. Finally, since the resulting paragraphs were relatively independent, they were themselves reduced into an article by simply concatenating them[1].

We used this approach to create five articles about New York City. Articles cost an average of $3.26 to produce, and required an average of 36 subtasks or HITs, each performed by an individual worker. Partition-workers identified 5.3 topics per article in the partition step. The average number article included 658 words. A fragment of a typical article is shown in Figure 2; this article consisted of 955 words and 7 sections: brief history, getting there, basic layout, neighborhoods, getting around, attractions and ethnic diversity. It was completed via 36 different HITs for a total cost of $3.15.

To verify the quality of these collaboratively written articles, we compared them to articles written individually by workers and to the entry from the Simple English Wikipedia on New York City [28]. To produce a comparison group of individually written articles, we created eight HITs which each requested one worker to write the full article. To control for motivations associated with reward, we paid these individuals $3.05, approximately the same amount as the average group payment. The resulting articles consisted of an average of 393 words, approximately 60% the length of the collaborative written articles.

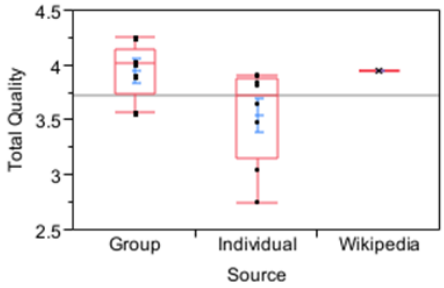

Figure 3: Rated quality of articles about New York City produced by Mechanical Turk workers acting individually or as a group using our framework compared to the quality of the same article on the Simple English Wikipedia.

We then evaluated the quality of all articles by asking a new set of workers to each rate a single article based on four dimensions: use of facts, spelling and grammar, article structure, and personal preference. Fifteen workers rated each article on five-point Likert scales. We averaged the ratings of the 15 raters across the four dimensions to get an overall quality score for each article.

On average the articles produced by the group were of higher quality than those produced individually (see Figure 3: mean quality for group-written articles = 4.01 versus 3.75 for individually-written ones, $t(11)=2.17$, $p=.05$). The average quality for the group-written articles was roughly the same as the Simple English Wikipedia article (Wikipedia quality=3.95). Not only was the average quality of the group articles higher than the individually written ones, but as Figure 3 also shows, the variability was lower as well ($t(11)=-2.43$, $p=.03$), with a lower proportion of poor articles.

Overall, we found that using CrowdForge to crowdsource the complex and interdependent task of article writing worked surprisingly well. Despite the coordination requirements involved in managing and integrating the work of dozens of workers, each contributing only a small amount of work, the group-produced articles were rated higher and had lower variability than individual-produced articles -- even though individuals were paid the same amount as the whole group and did not have to deal with coordination challenges -- and similar in quality to corresponding Simple Wikipedia articles.

Quality Control

In the above study each partition task was completed by a single worker. This creates the possibility that a single bad partition (i.e., outline) could have a large negative effect on the whole task. We found this did occur in one of the group articles, with a bad outline's effects cascading down the task chain. It is remarkable that despite this brittleness, we still found a robust advantage of the group condition over the individual condition, speaking to the strength of the

[1] For other kinds of articles there could be another crowdsourced reduce phase that integrates the paragraphs.

approach. However, in many cases we would like to minimize the likelihood of any task failing due to a single low quality worker by combining multiple workers' results.

Our approach to dealing with this challenge is to utilize additional Map or Reduce tasks to supporting fault tolerance and quality control. For example, to represent Castingwords transcription flow, described earlier, in the CrowdForge framework, workers verifying the results of other workers' outputs can be represented as a Map task that applies a verification function to each value. Other kinds of quality control processes can also be applied; for example, voting on the best choice can be represented as a Reduce task in which a single output is chosen from multiple workers' outputs based on the vote. Other kinds of human intelligence tasks could also be used, such as a Reduce task in which workers combine the best aspects of other workers' outputs rather than choosing a single best output. An advantage of this approach is that quality control steps are treated the same as other kinds of subtasks, minimizing added complexity.

A particularly interesting question is whether more complex quality control methods that require human intelligence -- such as combining the best aspects of multiple workers' outputs -- would work better than methods such as simple voting. Merging results (in this case, article outlines) could have a number of advantages over voting. The likelihood of a poor outline could be reduced, since at the very least – if workers did no merging at all – the best of the outline options should be chosen. It is also possible that the merged outlines could be better than the initial outlines, if the best aspects of each of multiple outlines were combined, or if seeing multiple outlines at once facilitates comparison between them and thus leads to better outlines.

To test these hypotheses we ran an experiment on quality control of article outlines. In the first phase we asked 20 workers to each independently generate an outline for an article on the recent Gulf of Mexico oil spill using the same procedure as in the article writing case study above. We then took these 20 outlines and randomly assigned them to 20 different sets of three outlines (outlines could be in more than one set). Each set was given to a different worker, who was asked to create a new outline for the article using elements from the 3 outlines in his or her set (i.e., a Reduce task). Workers were instructed to use any elements from any of the outlines they had available, but were not allowed to add new elements. This resulted in 20 merged outlines.

For evaluation we crossed the 20 initial and 20 merged outlines, and asked workers to choose which outline would result in a better article. To ensure that workers evaluated both outlines, they were also required to identify matching elements in the two outlines, following best practices outlined in [14]. Merged outlines were rated higher than the initial outlines: 61% of merged outlines were chosen compared to 39% of initial outlines. A binomial test revealed the difference in choice preferences were statistically significant ($p < .001$). Histograms of choice preferences for individual outlines are shown in Figure 4. Merged outlines also had fewer poor outlines: while 7 of the initial outlines were preferred 35% or less of the time, no merged outlines had preference values lower than 35%. Furthermore, the best merged outlines were considered better than the best initial outlines: the best initial outline was preferred 74% of the time, while 3 merged outlines were preferred more than that with the highest preferred 90% of the time. Together these results suggest that complex quality control tasks such as combining the results of multiple workers' outputs can be more effective than simple voting. CrowdForge makes such quality control tasks simple to implement as Reduce tasks.

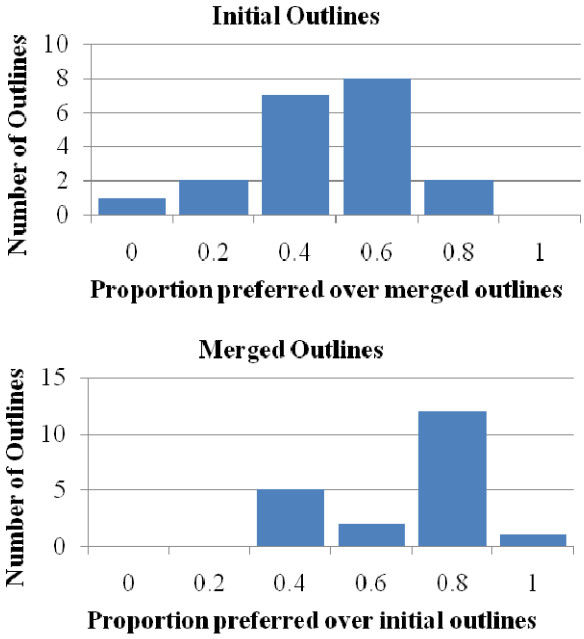

Figure 4. Histograms of participants preference choices for initial and merged outlines.

Researching a purchase

We also investigated a different task— researching purchase decisions—in order to test the generality of the framework. Specifically, we applied our framework to commission decision matrices, in this case to help consumers compare automobiles. This example extends the framework by showing how one can partition the initial task on multiple dimensions (or, equivalently, repartition each element of the original partition). In the partition HIT for this problem, one worker was given a short description of a consumer (a hypothetical suburban family that drives their two children to and from school and enjoys occasional road trips) and asked to submit criteria they would evaluate a car on (e.g., reliability, safety). Another worker was given the same description and asked to submit a list of potential competitors (e.g., Honda Odyssey, Ford Escape). Combining the resulting lists yielded a matrix resembling a product comparison table. In the map step, workers were asked to submit facts for one cell in the table, for example evidence

relevant to safety ratings for the Honda Odyssey they might find from an online review of the car. Finally, in the reduce step workers were given all the facts for a cell collected by workers in the map step, and were asked to write a single sentence consolidating them.

The entire task was completed in 54 different HITs for a total cost of $3.70. An excerpt from the resulting product comparison table is shown in Figure 5. We had no success getting even a single worker to generate a similar product comparison chart individually, even when offering more money than we paid the entire group, suggesting some complex tasks may not be amenable to completion in micro-task markets without appropriate decomposition.

PROTOTYPE

We implemented a software prototype to test our approach by allowing task designers to indirectly use MTurk to solve complex problems. It was this prototype that generated the HITs used in the experiments described above. The prototype allows task designers to break complex problems down into sub-problems, to specify the relationship between the sub-problems, and to generate a solution using MTurk. The system consists of a web user interface for the task designer, and a backend server which interfaces with Amazon's MTurk servers. The web user interface in Figure 6 allows users to define each step in the problem solving process and to specify the flow between each step. The server-side component creates MTurk HITs, consumes their results, and generates new HITs as needed. The prototype is written in Python using Django [8], a high-level web framework for rapid application development. Boto [3], a Python interface to Amazon Web Services, is used to communicate with MTurk. Source code for the prototype is available at https://github.com/borismus/crowdforge.

The system abstracts the entire process as a *problem*, which tracks the state of the current complex task. A problem references multiple *HIT Template*s (which may be either partitions, maps, or reduces), and a *flow* that defines the dependencies between the HIT Templates. The prototype allows multiple problems to exist in parallel, each one tracking its own currently active HIT Template. HIT Templates are parameterized templates used to create HITs on MTurk, specifying basic parameters like title, HTML body and compensation amount. Finally, flows manage the sequential coordination between HITs, as well as transferring data between HITs. When a user creates a new problem, they specify which flow to use to solve that problem.

	Honda Odyssey	Ford Escape	Nissan Pathfinder
Safety Rating	Honda's Odyssey scored 5 stars for front and side impact in tests conducted by the NHTSI, and features anti-lock brakes for added stability in turns when braking.	Ford's Escape model scored high marks in front and side crash testing, with a safety rating of 9.9 and pretensioner seatbelt feature that tightens automatically in a crash.	Nissan's Pathfinder gets high marks for safety in NHTSI tests and scored 4 stars for front crash test and is offered in a couple different body styles.

Figure 5: An excerpt from the product comparison table

Flows are implemented as python classes which have the **on_stage_completed**(self, stage): method. When all of the hits for the given hit type have been completed, CrowdForge calls the problem's flow's **on_stage_completed** method. Users can create a new flow by implementing a subclass of **crowdforge.flows.Flow** and registering it with **crowdforge.flows.register**. CrowdForge comes with several pre-built flows, such as the **SimpleFlow** which supports the simple partition-map-reduce scenario we used in generating NYC articles, and partition selection and verification flows that include additional verification steps.

The system uses a notification-based flow control mechanism to manage which tasks and templates are posted. Every few minutes the system monitors active problems for four kinds of events, and fires notifications as needed. The *result retrieved* notification fires when the system detects a new result from MTurk. The *HIT expired* notification fires when a HIT that was posted by the prototype expires due to the HIT lifetime running out. The *HIT complete* notification fires when all instances of a HIT were completed by workers. The *stage complete* notification fires when all HITs of the currently active HIT Template are completed or expired. Often, steps in CrowdForge require user submitted data from previous steps. Since CrowdForge is constantly polling MTurk to get newly submitted results, we have all of the results from previous submissions in the Results table in the database. All of these results are linked to the hit, and the hit type. The flow code can fetch these results using django's object relational model API.

For example, the simplest predefined flow (Figure 1) starts with a partition HIT Template, the result of which is fed into a map HIT Template, the results of which are fed into a reduce HIT Template. Transitions between these three HIT Templates occur when the s*tage complete* notification fires. In the article writing example, this flow takes the article outline generated by a worker completing a partition HIT, and creates map HITs to collect facts for each heading in the outline. Note that this process is dynamic: the number of headings does not need to be specified beforehand by the task designer. Once all map HITs for a heading are complete, the flow posts a reduction HIT to consolidate all facts collected in the map HITs into a paragraph.

COMPLEX FLOWS

The example of article writing assumed a simple linear flow from partitioning to mapping to reduction, which may not be powerful enough to represent some tasks. For more general cases, subtasks can be themselves be broken down into partition, map and reduce phases (Figure 7). For example, in journalism, writing headlines and leads and soliciting quotes themselves can be broken into parallelizable partition, map and reduce phases. The notification-based architecture of the CrowdForge prototype allows this kind of nesting to be implemented as a custom flow (future work will allow the creation of nested flows using GUI tools).

Below, we discuss a case study involving a much more complex flow.

Crowdsourcing Science Journalism

We chose to address the complex problem of crowdsourcing the science journalism process; i.e., turning a paper published in an academic venue (such as *Science*) into a newspaper article for the general public. This task is challenging for a number of reasons. Just reading a complete academic paper may require more motivation and expertise than any single crowd worker possesses, let alone any subsequently writing. However, seeing only a portion of the article might not provide workers with the whole story; thus, it is unsuitable for the simpler partitioning used in the encyclopedic article case study. Furthermore, a science article has a particular structure to it that the crowd output would need to adhere to; enforcing this structure provides additional constraints. Finally, the task may simply require more expertise than available in the task market. Thus we chose this task in the spirit of a "grand challenge", hoping that even if the process failed we would learn about where the CrowdForge approach would break down.

To help us shape this the task we partnered with two professional journalists interested in the question of whether science journalism could be crowdsourced. We started by choosing an article published in *Science* examining the role of social influence in determining popularity in an artificial music market [22]. Although the research was complex and moderately technical, we believed that crowd workers would be able to understand it.

We worked with the journalists to identify a typical structure for a popular science article, including creating a news lead, describing what scientists did, what they found, getting a quote from a relevant expert and an author of the study, and describing implications and future work. For each of these sections we worked to develop subflows that would produce them. Some of the subflows required iteration and trying several different approaches. In total our

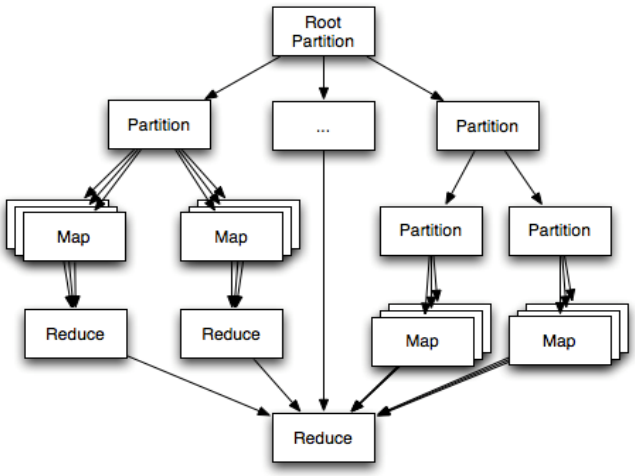

Figure 7. Nested subtasks forming a complex flow.

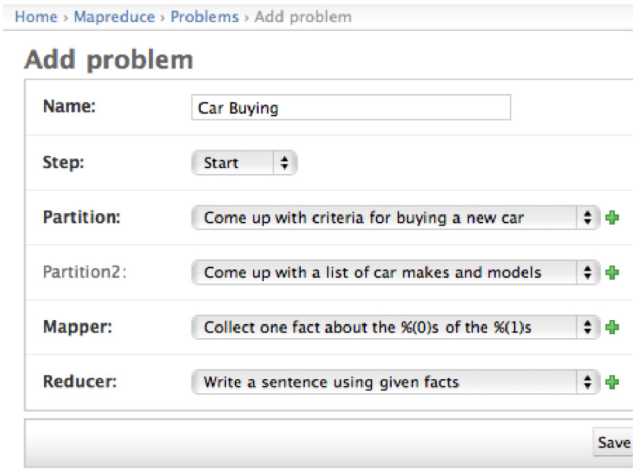

Figure 6: Creating a problem with the web user interface.

article generation task involved 11 subflows comprising 31 subtasks, which represent 262 worker judgments and a total cost of approximately $100. To give an idea of the tasks involved we will describe two interesting subflows in more detail below: generating a news lead and describing what the researchers did.

Generating a news lead requires quickly and succinctly conveying what the article is about in an engaging way that draws the reader in. To do this we initially provided workers with the article and asked them to tell us the single most important thing that a general reader needs to know about the finding in the paper. However, this task proved to be too large a chunk for workers to complete. Through several iterations we discovered task characteristics which made the process more amenable to workers. First, we used a "consolidate" process pattern in which the results of a large set are consolidated down to an input that better matches the limited attention profile of the worker. A simple example is presenting workers with the abstract of a scientific paper, in which the article authors have already consolidated the paper the paper down to the key points; however, we used various forms of consolidation throughout the process, including generating tags describing key terms from the abstract (to help find relevant experts) and extracting pieces of the article for different tasks (in the description of research, discussed below). Second, we provided workers with concrete examples of what we desired from them, e.g., in the form of an abstract and news lead from an exemplar article. This "exemplar" pattern proved particularly useful throughout many tasks, as mapping the sample input and output to the target article made the desired outcome much clearer than instructions could.

We used similar task patterns ("consolidation" + "exemplar") in the flow for generating a description of the research procedures, although with a slightly different instantiation. First, a partition task asked workers to extract the sections from the article that corresponded to different experiments. Then, for each experiment workers were asked

to summarize what the researchers actually did, again providing them with a sample experiment and description.

Evaluation

To evaluate the quality of the resulting items we enlisted experts with complementary skills, including a professional journalist, the first author of the *Science* paper, a graduate student doing research on social computing, and one of the authors of this study. Each rater was given a survey asking them to rate each of 16 news leads and 8 research descriptions on a 7-point Likert scale. Items were arranged in one of two randomized orders to control for order effects. Raters were in moderate agreement across items (alpha = .74).

Overall, the results were surprisingly good. According to the author of the paper, "it was a bit below what you would see in a high-quality publication like the NY Times, but the best were not totally different (although the worst were pretty bad)." The professional journalist had a similar reaction: "I'm really impressed by the quality of the answers. The abstract from the Salganik paper is not that technical by the standards of scientific literature, but the key news point -- that social influence helps determine success -- is contained in just one line. Yet 7 of the 10 workers who did the task put that point in their lead."[2] Ratings for the news leads ranged from 6.25 to 1.25 (higher being better), with a standard deviation of 1.3. Research descriptions fared less well, ranging between 4.4 and 1.8 (*SD* = .93). To provide some insight into the range of content created, the best rated (6.25) and worst rated (1.25) news leads are shown below:

> "Blockbusters, bestsellers, hit songs - the main variable that makes these things more popular than their lesser-known counterparts is not quality differences, according to a recent study. The answer lies in social influence - the more people know a certain movie is the one to watch, the more likely it will become popular." (Best)

> "The psychology of song preference. Song quality isn't the best predictor of song success." (Worst)

As indicated in the quotes above the results, while good, were not perfect. The resulting article would not be likely to win a Pulitzer, for example, which is not surprising given that workers were not trained nor expert writers. This suggests their output might be best used as part of a work flow in which professional editors or writers subsequently massage the crowdsourced work. Their value in this regard was supported by the journalist, who said: "An editor does not require that a reporter (or in this case a Turker) produce beautifully polished copy. But editors do need to be confident that the copy captures the main points of the story, because editors don't expect to have to do any additional reporting themselves. From that point of view, the replies are impressive. The quality of the writing is variable and in some cases a little muddled, but it is more important to note that many of the workers produced leads that captured the newsworthy element of the paper."

Perhaps more concerning, none of the leads or descriptions mentioned the important finding that there was high unpredictability about which items became popular. However, upon discussion the article author noted that "lots of the 'real' reporters missed the unpredictability part too", while the journalist raised the point that this might actually be due to differences in audience rather than missing the point: "the disagreement arises because the two are writing for different audiences: a journalist for lay readers and an academic for other researchers. The two audiences often care about different things."

Together these results suggest that despite the lack of expertise, limited time and effort, and limited context provided by crowd workers, assembling the output of many small judgments through the proper coordination can result in highly complex artifacts of surprisingly good quality.

LIMITATIONS

While we have demonstrated some of the strengths of the CrowdForge approach, here we discuss some limitations. First, the system currently does not support iteration or recursion, requiring the task designer to specify each stage in the task flow (though workers can define their own partitions and thus what future tasks will be). While CrowdForge is compatible with iteration, the strength of existing crowdsourcing toolkits that focus on iteration (e.g., TurKit) led us to focus our contribution elsewhere. We plan to merge both approaches in future work; this would enable, for example, one stage (e.g., a vote) to determine if another (e.g., a further partition) is necessary.

Second, and more fundamentally, CrowdForge is based on the idea that complex work can be broken up into relatively small and independent pieces with the system managing the coordination dependencies between those pieces. However, these assumptions can be violated. For example, it is possible that some work may not be easily decomposable into units small enough to match the task capacity of the workforce. We approached these limits with the science journalism task: although we had workers extract each experiment in the article separately to minimize the work, they still needed to read and summarize an entire experiment in order to complete the task; had the experiments been larger or more complex this step would not have succeeded as well. Another possibility is that the decomposition and recomposition of tasks, along with necessary intermediate quality control steps, could introduce more overhead and cost than they are worth. For example, it is likely not worth decomposing an article into independent sentences and requiring workers to merge arbitrary sentences into paragraphs. Finally, in situations where tasks really cannot be decomposed, selecting another task market where individuals have higher skill or motivation (e.g., oDesk, TopCoder) may be necessary.

[2] This quote was from a time when only 10 leads were collected.

Another violation of the assumptions may occur if subtasks are not independent. For example, while the diversity provided by multiple judgments was sufficient for the map phase of the New York City article to succeed, CrowdForge provided no guarantee that workers would not select diverse information. Providing them information about other workers' outputs could be valuable in preventing workers from all returning the same results. However, the advantages of providing additional context need to be weighed against the costs of increased task load (since workers need to process more information) and the potential for seeing other workers' judgments leading to negative consequences such as cognitive tunneling, information cascades, and production blocking [8][29]. This suggests that in some contexts (e.g., brainstorming) independent generation and subsequent merging of output may be superior to iteration on others' work, which in others exposure to others' outputs could improve the final product as workers could build on each others' work or reduce the bias of an individual. Understanding the appropriate times and ways to provide context and visibility into the work of others is an important area for future crowdsourcing research.

CONCLUSION

In this paper we presented a general-purpose framework for solving complex problems through micro-task markets. The framework manages the coordination between workers so that complex artifacts can be effectively produced by individuals contributing only small amounts of time and effort. Based on concepts from coordination science and distributed computing, the CrowdForge framework provides a systematic and dynamic way to break down tasks into subtasks and manage the flow and dependencies between them.

We demonstrate through three case studies and multiple experiments how the framework can break down complex tasks such as writing an article or researching a purchase decision into flows of partition, map, and reduce subtasks. In the article writing case we showed that CrowdForge-produced articles were rated more highly and had lower variability than individual-produced articles, despite the coordination requirements of managing and integrating dozens of workers, and were rated of similar quality as Simple Wikipedia articles. In an extension to this example, we showed how to insert a quality control step in the flow and demonstrated the value of combining the best aspects of multiple workers' outputs rather than simple voting. In the purchase decision case we were unable to get even a single individual to complete the task given the high effort involved, but could accomplish the goal using CrowdForge with low monetary costs. Finally, we explored more complex flows and subtasks using the domain of science journalism, demonstrating how a popular press article could be created despite crowdworkers having limited expertise, time and effort, and context. In this example we also discuss useful patterns for subtask design, such as the "consolidate" and "analogy" patterns. Finally, we discuss limitations of and fundamental challenges for the approach.

Furthermore, as the nature of work itself becomes more distributed, such an approach has the potential to change the way that work gets done, enabling many more people to be involved in solving complex problems ranging from business intelligence to writing software. However, markets don't work well for complex tasks when the employer cannot define exactly what they want in advance or if the contract is difficult to pre-define. The CrowdForge framework reduces this need for predefinition by allowing for subtasks to be dynamically generated by the market itself. It also supports scaling up complex tasks to involve many workers by managing the coordination dependencies between them.

As general purpose markets continue to evolve, there is a growing need to be able to solve a wider range of tasks of increasing complexity and coordination requirements. Although our framework draws from ideas in distributed computing such as MapReduce, using humans instead of machines as processors provides both distinct benefits and challenges. For example, our human-driven partition step is novel and unique to human computation. However, the unreliability and unknown expertise of each worker necessitates more complex and nested flows for quality control. Here we provide a conceptual framework and vision that we hope will inform and inspire future researchers and task designers in taking on more complex and even "grand challenge" tasks that will push the limits of what can be crowdsourced.

There are a number of directions we are exploring for future work. Most immediately, one challenge is extending our GUI to support more complex, nested flows so that task designers with no programming experience can complete arbitrarily complex work that involves high coordination dependencies. Looking further ahead, we are interested in exploring the possibilities of the CrowdForge framework in very different kinds of task markets. Although in this paper we implemented systems for two different platforms (MTurk and CrowdFlower), both have similar worker characteristics (indeed, many CrowdFlower tasks are posted on MTurk). If the framework was applied to a market in which the expertise of individual workers was better known (e.g., in a corporation) there might be greater opportunities for managing resource allocation of workers to appropriate tasks. Feedback about the selection and quality of their past work could also be useful for improving resource allocation if the system had a shared memory of individual workers' history across tasks.

ACKNOWLEDGMENTS

This work was supported by NSF grants OCI-0943148 and IIS-0968484, and the Center for the Future of Work, Heinz College, Carnegie Mellon University.

REFERENCES

[1] Bal, H.E., Steiner, J.G. and Tanenbaum, A.S. Programming languages for distributed computing systems. ACM Computing Surveys (CSUR) (1989) vol. 21 (3).

[2] Becker, G.S. and Murphy, K.M. The division of labor, coordination costs, and knowledge. The Quarterly Journal of Economics (1992) vol. 107 (4) pp. 1137-1160

[3] Boto, http://code.google.com/p/boto/

[4] Callison-Burch, C. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1* (2009) pp. 286-295

[5] Cole, F., Sanik, K., DeCarlo, D., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S., and Singh, M. How well do line drawings depict shape? In *ACM SIGGRAPH* (2009), 1–9.

[6] Dean, J. and Ghemawat, S. Map Reduce: Simplified data processing on large clusters. *Communications of the ACM*, 51, 1 (2008), 107-114.

[7] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L. "ImageNet: A large-scale hierarchical image database," cvpr, pp.248-255, *IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[8] Diehl, M. and Stroebe, W. Productivity loss in brainstorming groups: Toward the solution of a riddle. *Journal of personality and social psychology* 53, 3 (1987), 497–509.

[9] Django, http://www.djangoproject.com

[10] Ipeirotis, P. (2010, Mar 3). The New Demographics of Mechanical Turk. http://behind-the-enemy-lines.blogspot.com/2010/03/new-demographics-of-mechanical-turk.html. Retrieved 9-21-20

[11] Isard, M., Budiu, M., Yu, Y., Birrell, A., and Fetterly, D. Dryad: distributed data-parallel programs from sequential building blocks. *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, (2007), 59–72.

[12] Heer, J. and Bostock, M. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of CHI* (2010). ACM, New York, 203-212.

[13] Horton, J. J., Rand, D.G. and Zeckhauser, R.J., The online laboratory: Conducting experiments in a real labor market (2010). NBER Working Paper Series, Vol. w15961.

[14] Kittur, A., Chi, E., Suh, B. Crowdsourcing User Studies With Mechanical Turk. In *Proceedings of CHI* (2008).

[15] Kittur, A., Lee, B., Kraut, R. E. Coordination in Collective Intelligence: The Role of Team Structure and Task Interdependence. In *Proceedings of CHI* (2009). ACM, New York.

[16] Kittur, A. and Kraut, R. E. Harnessing the wisdom of crowds in wikipedia: quality through coordination. In *Proceedings of CSCW* (2008). ACM, New York.

[17] Little, G., Chilton, L.B., Goldman, M., and Miller, R.C. Turkit: human computation algorithms on mechanical turk. In *Proceedings of UIST* (2010), ACM, New York, 57–66.

[18] Malone, T. and Crowston, K. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26, (1994), 87-119.

[19] Malone, T., Yates, J. and Benjamin, R. Electronic markets and electronic hierarchies. *Communications of the ACM*, 30 (6). 484-497.

[20] Mintzberg, H. 1979. The Structuring of Organizations.Prentice-Hall, Englewood Cliffs, N.J.

[21] Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, (2008), 1099–1110.

[22] Salganik, M.J., Dodds, P.S., & Watts, D.J. Experimental study of inequality and unpredictability in an artificial cultural market. *Science, 311* (2006), 854.

[23] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A.Y. Cheap and fast---but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics* (2008), 254-263.

[24] Skillicorn, D.B., Talia, D. Models and languages for parallel computation, *ACM Computing Surveys* 30(2) (1998) 123–169.

[25] Sorokin, A., Forsyth, D. Utility data annotation with Amazon Mechanical Turk, *First IEEE Workshop on Internet Vision at CVPR* (2008).

[26] Van de Ven, A., Delbecq, A. and Koenig, R. Determinants of coordination modes within organizations. *American Sociological Review*, 41 (1976), 322-338.

[27] Williamson, O. Transaction-Cost Economics: The Governance of Contractual Relations. *The Journal of Law and Economics, 22* (2). 233.

[28] Wikipedia Simple entry on New York City, http://simple.wikipedia.org/wiki/New_York_City

[29] Woods, D. & Cook, R. 1999. Perspectives on human error: Hindsight biases and local rationality. In F. Durso (Ed.), Handbook of applied cognition. NY: John Wiley.